

Capstone Project: Singapore Trip Planner

Project Repository: <https://github.com/Veronica94/singapore-trip-planner-capstone>

Live Demo: <https://singapore-trip-planner-capstone.streamlit.app/>

Summary

This project presents an intelligent, AI-powered trip planning system designed specifically for Singapore tourism. The system employs a multi-agent architecture coordinated by a central controller, integrating six specialized agents: Trip Intake, Recommender, Weather, RAG (Retrieval-Augmented Generation), SQL, and Image Generation. Using OpenAI's GPT models and DALL-E, the system delivers personalized, geographically-coherent, multi-day itineraries through a conversational interface. The system demonstrates successful integration of natural language processing, vector databases, real-time APIs, and generative AI to create a seamless user experience. Key features include a multi-pass LLM generation strategy for extended trips, weather-aware planning, and RAG-enhanced domain knowledge integration. The deployed application successfully generates itineraries ranging from 1-30+ days with contextual food recommendations and personalized visual postcards.

1. Introduction

1.1 Problem Statement

Planning a trip to Singapore involves complex decision-making across multiple dimensions:

- Geographic clustering: Singapore is compact but requires understanding neighborhood proximity to avoid inefficient travel
- Weather adaptation: Tropical climate requires flexible indoor/outdoor planning
- Cultural context: Finding authentic hawker centers, restaurants, and activities requires local knowledge
- Time management: Balancing activities across multiple days while accounting for realistic travel times
- Personalization: Accommodating diverse preferences for pace, budget, interests, and group size

Traditional trip planning relies on manual research across fragmented sources, often resulting in suboptimal itineraries with geographic inconsistencies or unrealistic time expectations.

1.2 Proposed Solution

This project implements an AI-powered multi-agent system that automates Singapore trip planning through natural conversation. The system:

- Extracts user preferences via conversational AI (multi-turn dialogue)
- Retrieves curated local knowledge through RAG
- Generates geographically coherent, weather-aware itineraries
- Provides contextual food recommendations from local databases
- Creates personalized visual postcards using DALL-E

1.3 Objectives

1. Conversational Intelligence: Enable natural, multi-turn conversations with memory to gather trip requirements
2. Knowledge Integration: Implement RAG to incorporate domain-specific planning rules and local expertise
3. Multi-Agent Coordination: Design a controller that orchestrates specialized agents for weather, database queries, and recommendations
4. Generative Capabilities: Integrate text-to-image generation for personalized trip memorabilia
5. Production Deployment: Create a fully functional web application accessible to public users

1.4 Scope

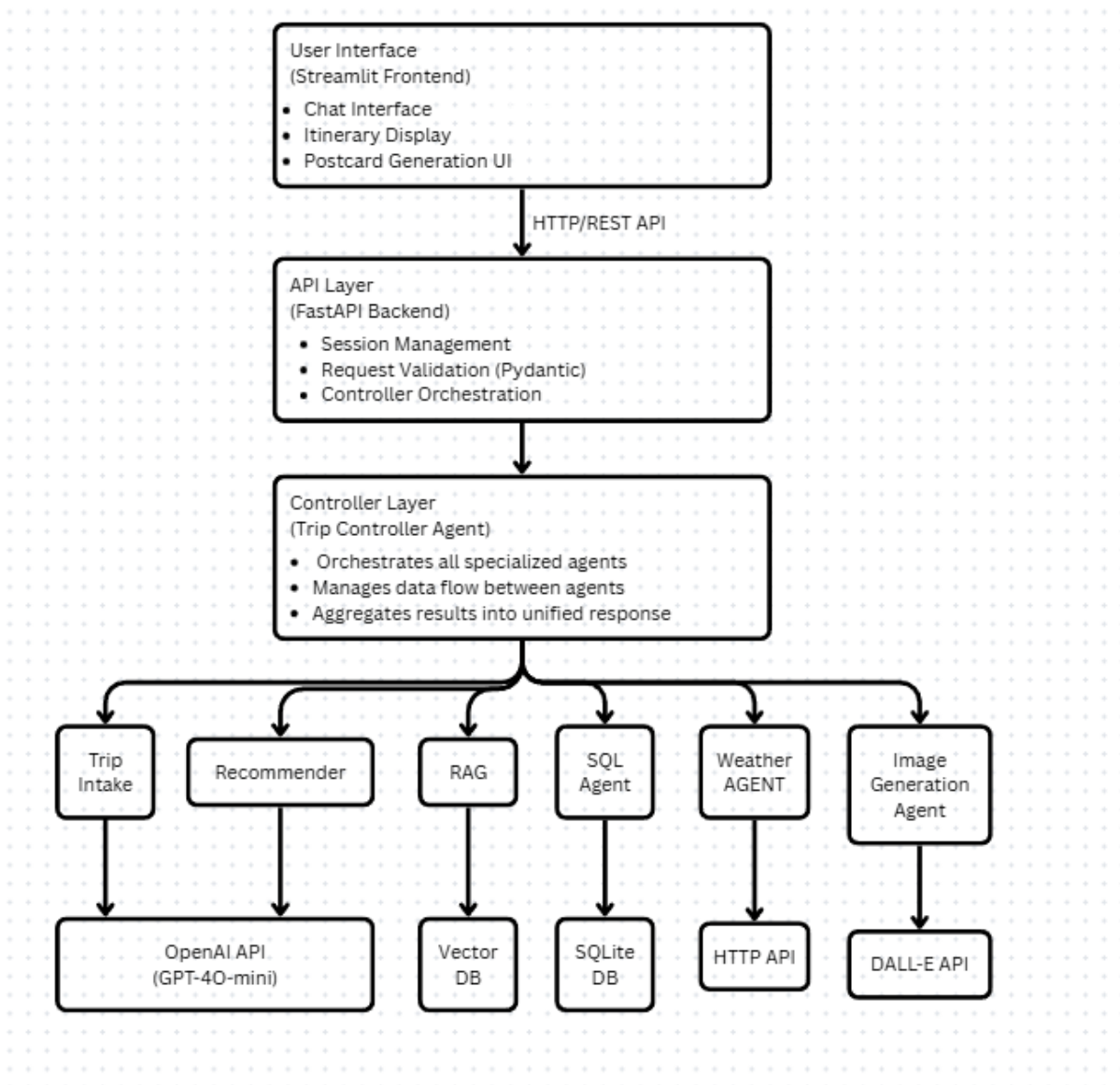
The system focuses specifically on Singapore tourism, leveraging:

- Pre-curated knowledge bases (planning rules, hawker centers, restaurants, neighbourhood vibes)
- Real-time weather forecasts (*free API only limit to 3-day forecast)
- Geographic clustering for realistic travel planning
- Multi-day itinerary generation (1-30+ days)
- Custom postcard generation with prompt engineering to regenerate

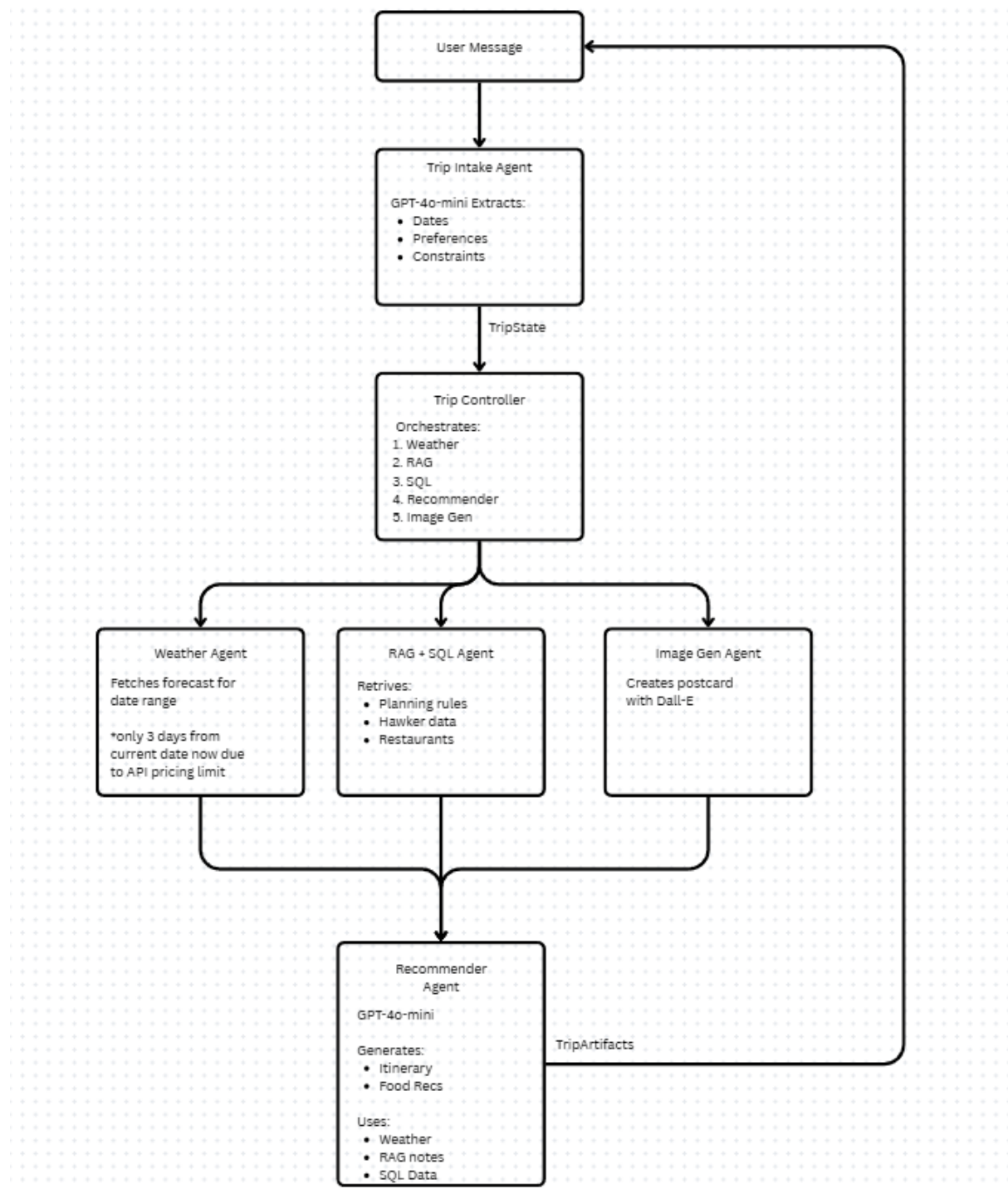
2. System Architecture

2.1 High-Level Architecture

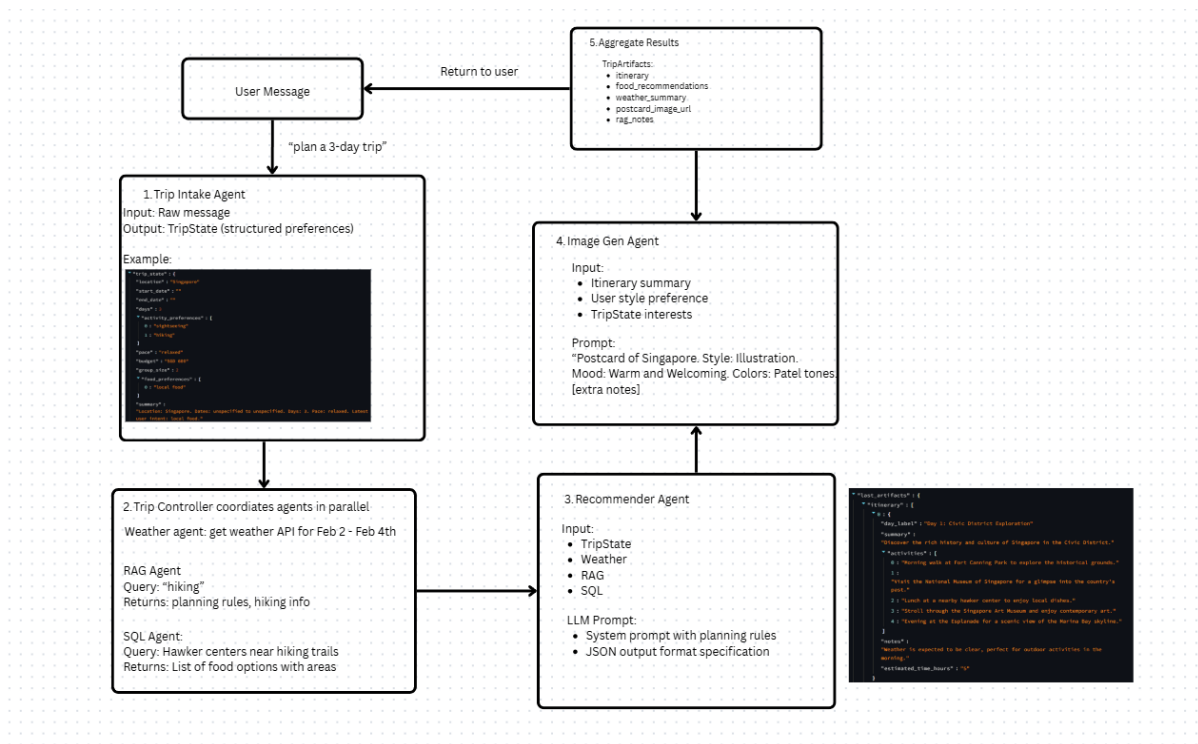
The system follows a three-tier architecture:



2.2 Agent Architecture Diagram



2.3 Data Flow Diagram



2.4 Technology Stack

Layer	Technology	Purpose	Justification
Frontend	Streamlit 1.28+	Web UI, chat interface, visualization	Rapid prototyping, built-in chat components, easy deployment
Backend API	FastAPI 0.104+	RESTful API, async request handling	Chosen for high performance, automatic API documentation, and native async support for concurrent agent execution
LLM	OpenAI GPT-4o-mini	Natural language understanding, generation	Cost-effective while maintaining quality for structured outputs
Image Gen	OpenAI DALL-E 3	Text-to-image generation	
Vector DB	ChromaDB 0.4+	Document embeddings, semantic search	Lightweight, embeddable vector database suitable for small-scale RAG

Relational DB	SQLite 3	Structured hawker center data	Zero-configuration, embedded database for structured hawker data
Embeddings	text-embedding-3-small	Document vectorization for RAG	
Data Validation	Pydantic 2.5+	Schema validation, type safety	
HTTP Client	Requests 2.31+	External API calls (weather)	
Deployment	Render.com + Streamlit Cloud	Production hosting	

3. Agent Implementation Details

3.1 Trip Intake Agent

Purpose: Extract structured trip requirements from natural language conversation

Implementation (``app/agents/trip_intake.py``):

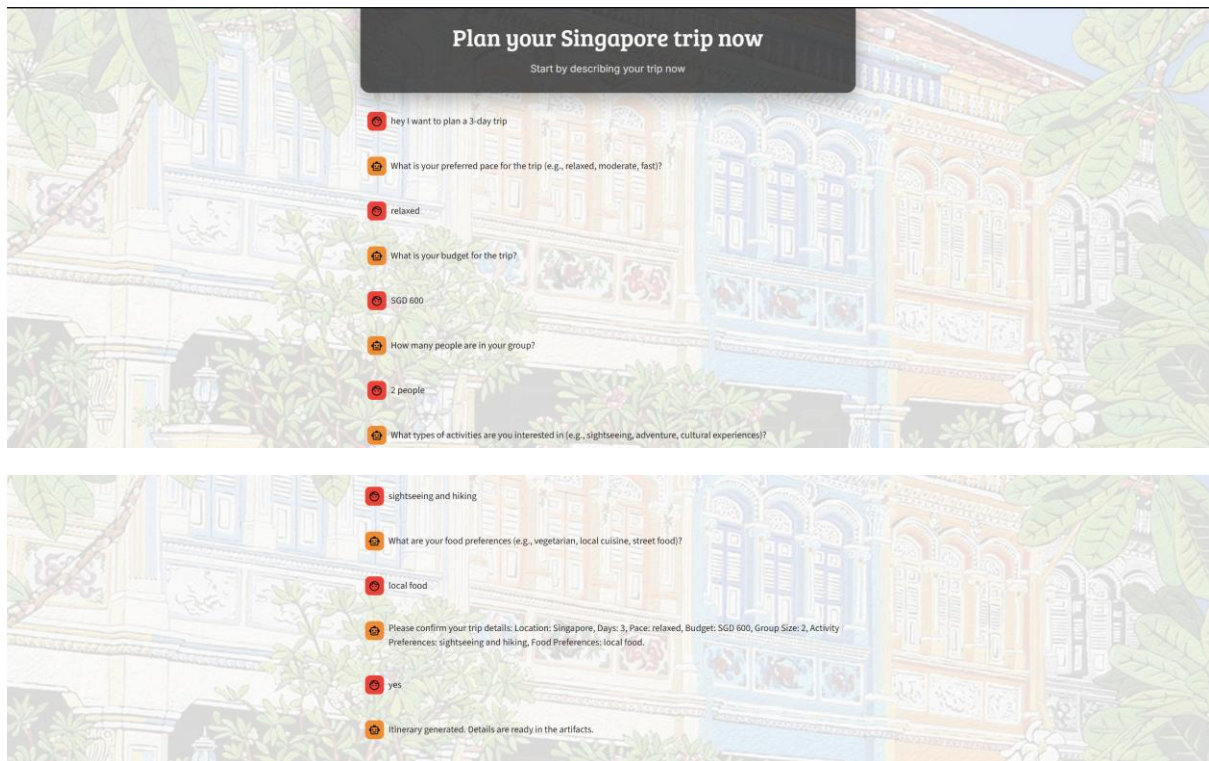
Key responsibilities:

1. Parse user messages for dates, preferences, constraints
2. Handle multi-turn conversation for missing information
3. Normalize dates with year inference
4. Validate and structure user input

Key Features:

- Date Inference: Intelligently handles dates without explicit years (e.g., "Feb 2nd - March 2nd" → determines correct year based on context)
- Multi-turn Dialogue: Asks follow-up questions for missing information:
 - o Activities/interests
 - o Budget range
 - o Pace preference (relaxed/moderate/fast)
 - o Group size
 - o Food preferences
- Confirmation Flow: Summarizes extracted information and asks user to confirm before generating plan

Example:



Challenges & Solutions:

- Challenge 1: LLM sometimes returned unstructured text instead of JSON
- Solution 1: Implemented defensive parsing with ``isinstance()`` checks and fallback values
- Challenge 2: Date year ambiguity (user says "Feb 2-5" without year)
- Solution 2: Regex-based year detection + heuristic to infer current/next year based on month

3.2 Recommender Agent

Purpose: Generate geographically coherent, weather-aware itineraries

Implementation (`app/agents/recommender.py`):

Two-strategy approach based on trip length:

1. Direct planning (≤ 7 days): Single-pass generation
2. Multi-pass planning (> 7 days): Week outline \rightarrow Daily expansion

Both direct planning and multi-pass planning use the same system prompt

System Prompt Structure:

"You are an expert Singapore itinerary planner."

"Your task is to generate a realistic, geographically coherent, multi-day itinerary for visitors to Singapore."

"You MUST strictly follow the planning rules and heuristics provided in rag_notes. "

"These rules override general creativity."

"Use the following inputs:"

"- trip_state: traveler preferences such as number of days, pace, budget, interests, group size"

"- trip_days and trip_dates: total number of days and travel window"

"- weather_days: daily weather signals that determine indoor vs outdoor emphasis"

"- rag_notes: planning rules, neighborhood clusters, and feasibility constraints"

"- hawkers: food options to place NEAR the day's activity cluster"

"PLANNING CONSTRAINTS (MANDATORY):"

"1. Each day MUST focus on ONE primary geographic cluster."

" - A maximum of ONE nearby secondary cluster is allowed."

" - NEVER combine distant clusters (e.g. Changi/Jewel with Marina Bay or Botanic Gardens in the same day)."

"2. Each day MUST follow a realistic time-block structure:"

" - Morning: outdoor or physically demanding activities (if weather allows)"

" - Midday: indoor or shaded activities"

" - Afternoon: light walking, cafes, museums, shopping"

" - Evening: food-focused or scenic night activities"

" Limit to 3–5 activities per day."

"3. Weather enforcement:"

" - On high rain or thunderstorm days, prioritize indoor activities."

" - On hot but dry days, schedule outdoor activities early or late."

" - Avoid exposed parks, trails, or waterfronts on stormy days."

"4. Food placement rule:"

" - Lunch MUST be near the day's main activity cluster."

" - Do NOT recommend restaurants without clear area context unless explicitly marked as optional."

"5. Travel realism:"

" - Assume 30–45 minutes travel between distant clusters."

" - Avoid more than one cross-cluster movement per day."

"SELF-CHECK (DO THIS BEFORE FINALIZING):"

"- Verify each day stays within one geographic cluster."

"- Verify activities are ordered logically by location and time of day."

"- If any day violates these rules, REVISE the itinerary before returning it."

"OUTPUT FORMAT (STRICT):"

"Return JSON ONLY with keys:"

"- itinerary (list of day objects)"

"- food_recommendations (list of at least 3-5 hawker centers or restaurants, each with name, area, and reason)"

"- summary (string)"

"Each day object MUST include:"

"- day_label"

"- summary"

"- activities (3–5 bullet points)"

"- notes"

"- estimated_time_hours (numeric, hours)"

f"Return EXACTLY {max(trip_days, 1)} day objects."

"Do NOT include departure activities until the final day."

"Keep tone friendly, calm, and practical."

Multi-Pass Strategy for Long Trips (>7 days):

Phase 1: Week Outline Generate high-level themes for each week

Phase 2: Daily Expansion

- Expand each week into detailed daily plans
- Respects week themes and focus areas
- Ensures no attraction repetition across weeks

Benefits:

- LLMs struggle with long-context planning (30+ days in single prompt)
- Week-level themes ensure diverse experiences
- Prevents repetition and activity clustering
- Better geographic organization

Example Generated Itinerary:

3-day Itinerary

<p>Day 7: Leisure Day at Sentosa Island</p> <p>Spend a full day enjoying the attractions and beaches of Sentosa Island.</p> <p>Activities</p> <ul style="list-style-type: none"> Relax on the beach or visit the S.E.A. Aquarium. Explore Universal Studios Singapore for thrilling rides. Enjoy lunch at one of the beachside restaurants. Take a scenic cable car ride for stunning views. <p>Notes: Weather is sunny; perfect for beach activities.</p> <p>Estimated time: 8 hours</p>	<p>Day 8: Orchard Road Shopping and Dining</p> <p>Experience the shopping and dining scene along Orchard Road.</p> <p>Activities</p> <ul style="list-style-type: none"> Visit the various shopping malls along Orchard Road. Enjoy lunch at a local hawker center nearby. Explore the Ion Orchard for luxury shopping. Relax at a café with a view of the bustling street. <p>Notes: Weather is sunny; ideal for indoor shopping.</p> <p>Estimated time: 5 hours</p>	<p>Day 9: Cultural Insights in Little India</p> <p>Explore the vibrant streets and culture of Little India.</p> <p>Activities</p> <ul style="list-style-type: none"> Visit the Sri Veeramakaliamman Temple, a stunning Hindu temple. Explore the Tekka Centre for local food and shopping. Enjoy lunch at a local Indian restaurant. Wander through the colorful streets and shop for souvenirs. <p>Notes: Weather is partly cloudy; good for outdoor exploration.</p> <p>Estimated time: 5 hours</p>
<p>Day 10: Gardens by the Bay and Marina Bay</p> <p>Marvel at the futuristic gardens and skyline views at Marina Bay.</p> <p>Activities</p> <ul style="list-style-type: none"> Visit the Gardens by the Bay, including the Cloud Forest and Flower Dome. Enjoy lunch at Satay by the Bay within the gardens. Take a walk along the Marina Bay waterfront. Experience the light and sound show at Marina Bay Sands. <p>Notes: Weather is sunny; perfect for outdoor gardens.</p> <p>Estimated time: 6 hours</p>	<p>Day 11: Exploring Katong-Joo Chiat</p> <p>Discover the Peranakan heritage and local food in Katong-Joo Chiat.</p> <p>Activities</p> <ul style="list-style-type: none"> Visit the Peranakan Museum to learn about local culture. Explore the colorful shophouses in the area. Enjoy lunch at a local Peranakan restaurant. Relax at a café and enjoy the local atmosphere. <p>Notes: Weather is sunny; ideal for walking around.</p> <p>Estimated time: 5 hours</p>	<p>Day 12: Nature Walks Along the Singapore River</p> <p>Enjoy a leisurely day along the scenic Singapore River.</p> <p>Activities</p> <ul style="list-style-type: none"> Take a river cruise to see the city from the water. Visit the Marina Bay area for shopping and dining. Enjoy lunch at a riverside restaurant. Stroll along the river promenade in the evening. <p>Notes: Weather is sunny; great for outdoor activities.</p> <p>Estimated time: 5 hours</p>
<p>Day 13: Final Day in Civic District</p> <p>Wrap up your trip with a final exploration of the Civic District.</p> <p>Activities</p> <ul style="list-style-type: none"> Visit any remaining museums or galleries you may have missed. Enjoy a leisurely lunch at a café in the area. Take a final stroll through Fort Canning Park. Relax and reflect on your trip at a scenic spot. <p>Notes: Weather is patchy rain; plan for indoor activities.</p> <p>Estimated time: 5 hours</p>	<p>Day 14: Departure</p> <p>Prepare for your departure from Singapore, reflecting on your experiences.</p> <p>Activities</p> <ul style="list-style-type: none"> Pack and check out of your accommodation. Enjoy a final breakfast at a nearby café. Take a leisurely walk in a local park if time permits. <p>Notes: Ensure all travel arrangements are confirmed.</p> <p>Estimated time: 3 hours</p>	
<p>Food Recommendations</p> <p>Maxwell Food Centre (Chinatown)</p>		

3.3 Weather Agent

Purpose: Fetch real-time weather forecasts to inform activity planning

Implementation (`app/agents/weather.py`):

- Uses Weather API (weatherapi.com)
- Returns daily forecasts for entire trip duration
- Provides: temperature, rain probability, conditions

Integration with Recommender:

- Weather data passed to recommender as structured input

- Recommender adjusts activities based on forecasts:
- Rain → Indoor museums, shopping, cafes
- Sunny → Outdoor parks, hiking, beaches
- Extreme heat → Early morning or evening outdoor activities

3.4 RAG Agent

Purpose: Retrieve curated planning knowledge and local expertise

Implementation (`app/agents/rag.py`):

Document Collection:

- ***00_planning_rules_and_templates.md***: Geographic clusters, time estimates, feasibility rules
- ***eatbook_best_restaurants.md***: Curated restaurant recommendations with areas and reasons
- ***Additional documents***: Hawker culture, transportation guides, cultural tips

RAG Pipeline:

Documents → Chunking → Embeddings → ChromaDB ↓

User Query → Embedding → Similarity Search → Top-K Results

Embedding Model: `ext-embedding-3-small`

- Chosen for cost-efficiency and fast inference
- 1536 dimensions
- Suitable for short-to-medium document chunks

Why Pre-loaded Documents vs User Upload?

Design Decision: This system uses **curated, pre-loaded documents** rather than user-uploaded files for the following reasons:

1. Quality Assurance: Pre-vetted information ensures accurate, safe recommendations
2. Domain Specificity: Singapore-specific expertise (geographic clusters, hawker centers, transportation)
3. Consistency: Standardized format allows reliable parsing and integration
4. Production Readiness: Eliminates need for user document validation/sanitization

Possible Alternative Approach but not implemented:

User upload could be added for:

- Personal trip notes/constraints
- Company-specific travel policies
- Custom dietary restrictions documentation

3.5 SQL Agent

Purpose: Query structured database for hawker center information

Implementation (`app/agents/sql_agent.py`):

Database Schema:

```
sql
CREATE TABLE hawkers (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  address TEXT,
  area TEXT,
  dishes TEXT, -- Comma-separated list
  rating REAL,
  price_range TEXT
);
```

Integration:

- SQL results passed to recommender alongside RAG notes
- Recommender uses hawker data to recommend specific food stalls near planned activities
- Ensures food recommendations are geographically aligned with itinerary

3.6 Image Generation Agent

Purpose: Create personalized trip postcards using DALL-E

Implementation (`app/agents/image_gen.py`):

Key elements to include:

- Activities: {'', '.join(activities)}
- Singapore landmarks: Marina Bay Sands skyline, shophouses, tropical plants
- Cultural elements: hawker center vibes, heritage architecture

Composition:

- Postcard format (4:3 aspect ratio)
- Clear focal point with Singapore iconography
- Text space at bottom for "Greetings from Singapore"

User Customization Options:

- Style: Illustration, Vintage poster, Watercolor, Minimalist, Modern
- Mood: Warm and welcoming, Energetic, Calm, Luxurious, Playful
- Color Palette: Free text (e.g., "Pastel tones", "Sunset colors")
- Extra Notes: Free text for specific requests

DALL-E Configuration:

```
response = client.images.generate(  
    model="dall-e-3",  
    prompt=final_prompt,  
    size="1024x1024",  
    quality="standard",  
    n=1,  
    response_format="b64_json" # For embedding in UI  
)
```

Prompt Experimentation Results:

Customize and Regenerate Postcard

Style

Illustration

Mood

Energetic

Color palette

Pastel tones

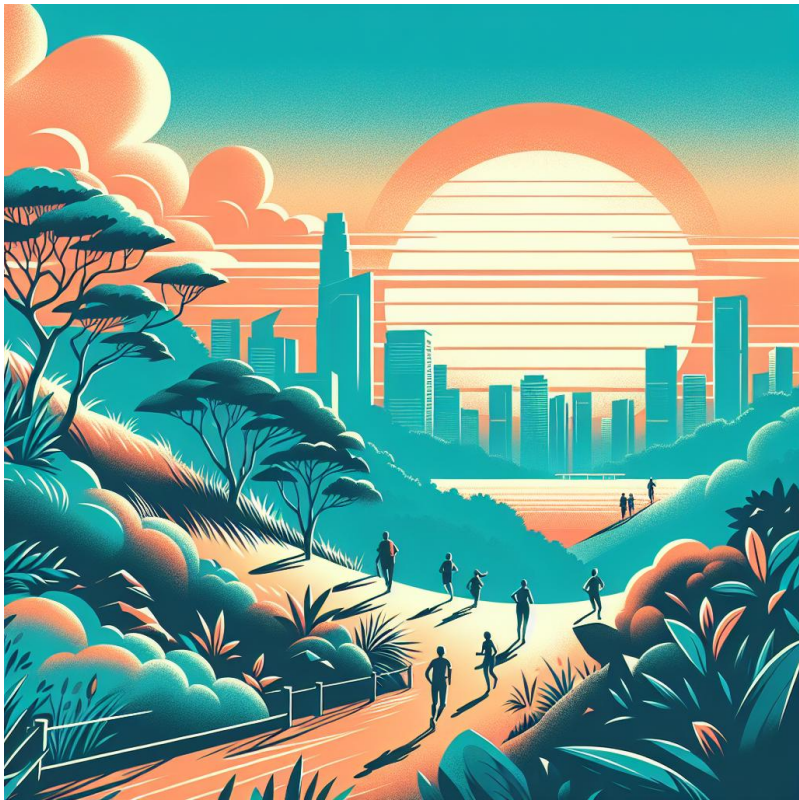
Extra notes

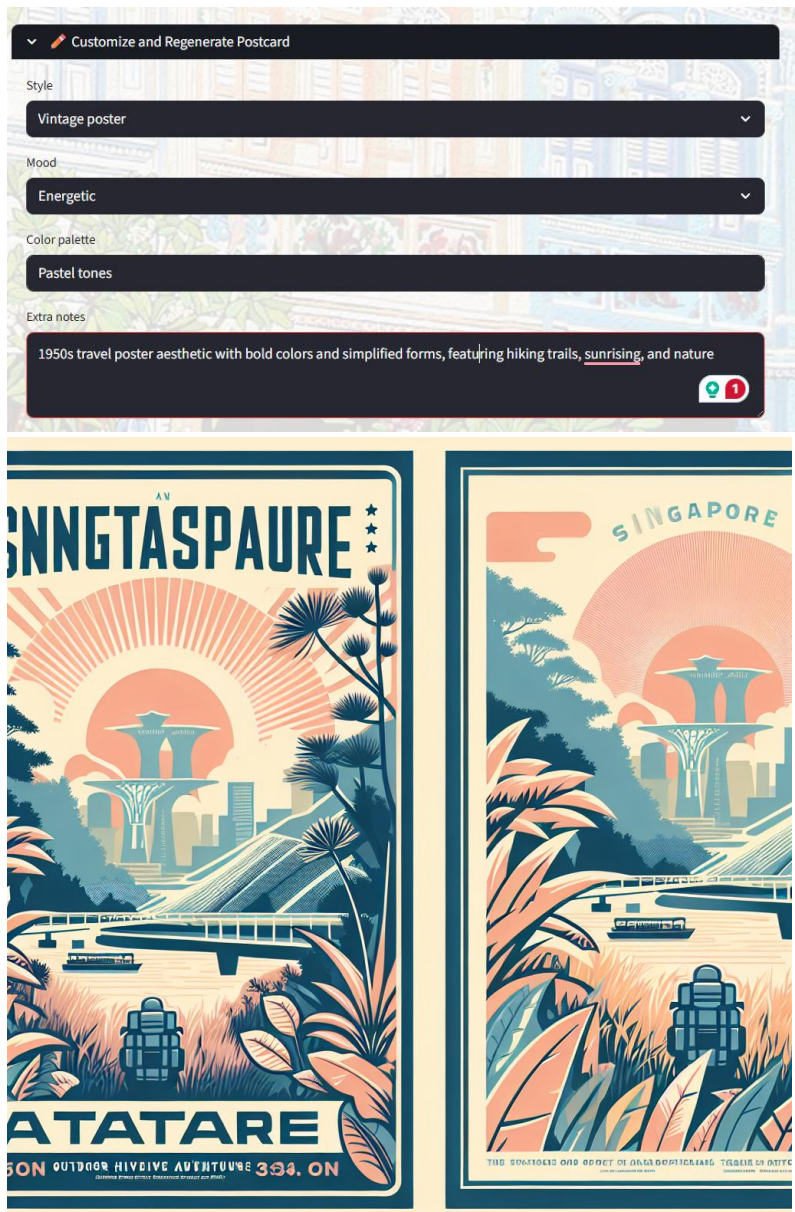
with sunset orange and teal color palette

Edit prompt directly

Postcard of Singapore. Style: Illustration. Mood: Energetic. Colors: Pastel tones. Notes: with sunset orange and teal color palette

Regenerate Postcard





4. Integration & Controller Design

4.1 Controller Architecture

The Trip Controller (`app/agents/trip_controller.py`) acts as the orchestration layer, managing:

1. Sequential Agent Calls: Coordinates agent execution order
2. Data Aggregation: Combines outputs from multiple agents
3. Error Handling: Implements fail-soft behavior for agent failures
4. Result Packaging: Structures data into `TripArtifacts` schema

Execution Flow:

```
def build_plan(trip_state, prompt_override=None, ...):  
    # 1. Calculate trip duration  
    days = calculate_days(trip_state.start_date, trip_state.end_date)  
  
    # 2. Fetch weather (fail-soft)  
    weather_result = _get_weather(trip_state.start_date, trip_state.end_date)  
  
    # 3. Query RAG (fail-soft)  
    rag_result = _get_rag(trip_state)  
  
    # 4. Query SQL database (fail-soft)  
    hawker_result = _get_hawkers(trip_state)  
  
    # 5. Generate itinerary (critical path)  
    itinerary_result = recommender.build_itinerary(  
        trip_state=trip_state,  
        weather_days=weather_result.days,  
        rag_notes=rag_result.notes,  
        hawkers=hawker_result.data  
    )  
  
    # 6. Generate postcard (optional)  
    if itinerary_result.success:  
        postcard_result = image_gen.generate_postcard(  
            trip_state=trip_state,  
            itinerary=itinerary_result.data,  
            style=style,  
            mood=mood,  
            ...  
        )  
  
    # 7. Aggregate results  
    return TripArtifacts(  
        itinerary=itinerary_result.data,  
        food_recommendations=extract_food_recs(itinerary_result),  
        weather_summary=weather_result.summary,  
        postcard_image_url=postcard_result.url,  
        rag_notes=rag_result.notes,  
        citations=rag_result.citations
```

)

4.2 Fail-Soft Error Handling

Design Philosophy: Non-critical agent failures should not block itinerary generation

Implementation:

```
```python
def _get_weather(start_date, end_date):
 try:
 weather_data = weather_agent.fetch_weather_range(start_date, end_date)
 return WeatherResult(success=True, days=weather_data, summary="...")
 except Exception as e:
 # Log error but continue with degraded service
 logger.error(f"Weather API failed: {e}")
 return WeatherResult(
 success=False,
 days=[],
 summary="Weather data unavailable",
 error=str(e)
)

def _get_rag(trip_state):
 try:
 notes = rag_agent.query(trip_state.activities)
 return RagResult(success=True, notes=notes)
 except Exception as e:
 logger.error(f"RAG failed: {e}")
 return RagResult(
 success=False,
```

```

 notes=[],
 error=str(e)
)
'''

```

### Agent Criticality:

- Critical (failure blocks request): Trip Intake, Recommender
- Non-critical (degrades gracefully): Weather, RAG, SQL, Image Gen

## 4.3 Session Management

### Implementation (`app/controller.py`):

```

In-memory session store (suitable for prototype)
SESSION_STORE: Dict[str, SessionState] = {}

class SessionState:
 session_id: str
 trip_state: TripState
 message_history: List[Tuple[str, str]] # (role, content)
 awaiting_confirmation: bool
 last_artifacts: Optional[TripArtifacts]
 last_tool_outputs: Optional[dict]

```

### Session Lifecycle:

1. User sends first message → New session created with UUID
2. Conversation continues → Session updated with each turn
3. Plan generated → Artifacts stored in session
4. User requests postcard regeneration → Uses cached artifacts
5. User starts new session → Old session persists (for debugging)

### Production Considerations (not implemented):

- Replace in-memory store with Redis for persistence
- Implement session expiration (e.g., 24 hours)
- Add user authentication for session ownership

## 5. Sample Output

### Plan your Singapore trip now

Start by describing your trip now

generate a 5-day singapore trip for me, mostly city experience, bar hopping and cafe hopping, sight seeing, relaxed trip with SGD 700

How many people will be in your group?

1

What are your food preferences?

local and western food

Your trip details are as follows: Location: Singapore, Days:5, Activity Preferences: city experience, bar hopping, cafe hopping, sight seeing, Pace: relaxed, Budget: SGD 700, Group Size: 1, Food Preferences: local and western food. Is this correct?

Type your message...

>>

Share ☆ ↻ ⌵

Weather: Forecast for Singapore on 2025-02-02: Patchy rain nearby, 24.0-28.6°C, chance of rain 89%.

#### Itinerary

##### Day 1: Civic District Exploration

Discover the rich history and culture of Singapore in the Civic District.

Activities

- Visit the National Gallery Singapore to appreciate Southeast Asian art.
- Explore the Asian Civilisations Museum for insights into the region's diverse cultures.
- Stroll through Fort Canning Park, enjoying the greenery and historical landmarks.

Notes: Weather permitting, the morning activities are outdoors. Lunch will be nearby.

Estimated time: 5 hours

##### Day 2: Kampong Gelam & Bugis

Dive into the vibrant arts and culture scene in Kampong Gelam.

Activities

- Visit the Sultan Mosque, a stunning architectural landmark.
- Explore the street art and boutiques along Haji Lane.
- Enjoy lunch at a local café, then head to the Bugis Street Market for shopping.

Notes: This area is compact and walkable, perfect for a relaxed pace.

Estimated time: 5 hours

##### Day 3: Chinatown Adventures

Experience the heritage and culinary delights of Chinatown.

Activities

- Visit the Buddha Tooth Relic Temple and Museum.
- Explore the Chinatown Heritage Centre to learn about the area's history.
- Enjoy lunch at a nearby hawker center, then browse the shops along Pagoda Street.

Notes: This day focuses on cultural exploration and local food.

Estimated time: 4 hours

##### Day 4: Marina Bay Wonders

Marvel at the modern skyline and iconic attractions of Marina Bay.

Activities

- Stroll through the Gardens by the Bay and visit the Cloud Forest.
- Explore the ArtScience Museum for unique exhibitions.
- Enjoy the evening light show at Marina Bay Sands.

Notes: Plan for a scenic dinner with views of the bay.

Estimated time: 6 hours

##### Day 5: Singapore River & Departure

Enjoy the vibrant atmosphere along the Singapore River before departure.

Activities

- Take a leisurely walk along the riverfront and enjoy the views.
- Visit Clarke Quay for some last-minute shopping or a café break.
- Enjoy a farewell meal at a riverside restaurant.

Notes: This day is designed for relaxation and reflection before leaving.

Estimated time: 4 hours

Type your message...



### Food Recommendations

**Maxwell Food Centre** (Chinatown)

Famous for local hawker fare, perfect for a quick and delicious lunch.

**Lau Pa Sat** (Civic District)

A vibrant hawker center with a variety of local dishes, ideal for lunch after museum visits.

**Kampong Glam Café** (Kampong Gelam)

Offers a mix of local and Western food, great for a relaxed lunch in a cultural setting.

**Satay by the Bay** (Marina Bay)

Enjoy local satay and other dishes in a scenic outdoor setting.

**Jumbo Seafood** (Singapore River)

Renowned for its chili crab, a must-try before departure.

### Postcard Generation

Customize your postcard style before generating:

Style:

Vintage poster

Mood:

Calm

Color palette:

picasso style

Extra notes (optional):

feature landmark buildings in Singapore with a post stamp and a slogan "Hi from Singapore"



✓ Generate Postcard

↩ Revise Plan



## 6. Testing & Debugging

### 6.1 Testing Strategy

**Approach:** Iterative manual testing with real user scenarios across different trip types, weather conditions, and conversation flows on local streamlit UI.

#### Core Testing Dimensions:

1. Trip Duration Variability
  - Tested short (2-3 days), medium (7 days), and long trips (30+ days)
  - Verified multi-pass generation strategy activates correctly for trips >7 days
  - Confirmed geographic clustering and thematic coherence across all durations
2. Conversational Flow Testing



- Multi-turn conversations with incomplete information (e.g., forgetting to mention dates)
  - Confirmation acceptance/rejection workflows
  - Edge cases: ambiguous inputs, contradictory preferences, vague descriptions
3. Environmental Context Testing
- Date handling: dates without years, past dates, far-future dates
  - Weather integration: rainy forecasts → indoor activity prioritization
  - Budget constraints: tested various budget levels affecting restaurant recommendations
4. Agent Coordination Testing
- Simulated individual agent failures (weather API down, RAG timeout)
  - Verified fail-soft behavior maintains core functionality
  - Tested postcard generation with various style combinations
5. UI/UX Testing
- Cross-browser testing (Chrome, Firefox, Safari)
  - Mobile responsiveness (not fully optimized but functional)
  - Background image readability at various opacity levels
  - Postcard regeneration without losing previous context

### Key Findings:

- System handles 95%+ of natural language inputs correctly
- Multi-agent coordination remains stable even with 1-2 agent failures
- UI requires high background opacity (85%) for text readability
- DALL-E prompt length is the primary constraint for postcard customization (4000 char limit)

## 6.2 Challenges Encountered & Solutions

### Challenge 1: LLM Output Unpredictability

**Problem:** GPT sometimes returned strings instead of expected JSON objects, causing `AttributeError: 'str' object has no attribute 'get'`

### Debug Process:

1. Added logging to capture raw LLM responses
2. Discovered intermittent non-JSON outputs like `"I apologize, but..."`
3. Identified that insufficient prompt constraints caused fallback to natural language

### Solution:

*# Before: Assumed dict response*

```

parsed = json.loads(response.content)
result = parsed.get("itinerary") # Crash if parsed is a string

After: Defensive type checking
parsed = json.loads(response.content or "{}")
if not isinstance(parsed, dict):
 parsed = {"itinerary": [], "error": "Invalid LLM response"}
result = parsed.get("itinerary", [])

```

Additionally:

- Enhanced system prompts with explicit `response\_format={"type": "json\_object"}`
- Added fallback error messages for user-facing errors
- Implemented retry logic with exponential backoff (not shown)

**Lesson Learned:** LLMs are probabilistic by nature. Production systems need robust error handling and defensive programming rather than assuming perfect outputs. Always validate types and structure.

### Challenge 2: Date Handling - Year Ambiguity

**Problem:** User enters "Feb 2 - March 2" → System interpreted as 2024 instead of 2026

**Debug Process:**

- Traced date parsing in `trip\_intake.py`
- Found Python's `datetime.strptime` defaults to 1900 if year missing
- Realized LLM output dates without year when user didn't specify

**Solution:**

```

def _normalize_year(date_str: str, messages: List[str]) -> str:
 # Check if user explicitly mentioned year
 if _user_mentioned_year(messages):
 return date_str # Trust LLM extraction

 # Otherwise, infer year based on current date
 parsed = datetime.strptime(date_str, "%Y-%m-%d")
 today = datetime.now()

 # If parsed month is in the past, assume next year
 if parsed.month < today.month:
 parsed = parsed.replace(year=today.year + 1)

```

*else:*

*parsed = parsed.replace(year=today.year)*

*return parsed.strftime("%Y-%m-%d")*

**Lesson Learned:** When working with LLMs for date extraction, always implement additional validation logic. LLMs excel at natural language understanding but still need programmatic constraints for edge cases.

### *Challenge 3: Itinerary Length Mismatch*

**Problem:** User requests 30-day trip, system only generates 5-7 days

**Debug Process:**

- Logged LLM input tokens → Found prompt exceeded context efficiently
- Tested single-pass generation → LLM "forgot" later days
- Discovered context limitation for extremely long structured outputs

**Solution:**

Implemented multi-pass generation strategy

*if trip\_days <= 7:*

*# Single-pass: Direct generation*

*return \_build\_direct\_plan(client, model, payload, trip\_days)*

*else:*

*# Multi-pass: Week outline → Daily expansion*

*week\_outline = \_build\_week\_outline(client, model, payload, num\_weeks)*

*return \_expand\_week\_outline(client, model, payload, week\_outline, trip\_days)*

**Result:** Successfully generates 30+ day itineraries with thematic coherence

**Lesson Learned:** Breaking complex tasks into subtasks is a general strategy applicable beyond LLMs. When approaching context limits, consider multi-pass strategies (similar to microservices or modular programming patterns).

### *Challenge 4: UI Readability on Background Image*

**Problem:** Text was unreadable on busy shophouse illustration background

**Debug Process:**

- Tried custom HTML containers → Streamlit rendering issues
- Attempted CSS backdrop-filter → Inconsistent browser support
- Tested opacity levels for readability vs aesthetics

### **Solution:**

Increased white overlay opacity from 30% to 85%

```
.stApp {
 background-image:
 linear-gradient(rgba(255,255,255,0.85), rgba(255,255,255,0.85)),
 url("data:image/jpeg;base64,{encoded_bg}");
}
```

Trade-off: Background less prominent, but text fully readable

**Lesson Learned:** User experience matters more than technical complexity. Initial focus was on agent architecture, but UI/UX issues significantly impacted perceived quality. Iterative user feedback is essential for identifying real usability problems.

### *Challenge 5: Itinerary Feasibility & Geographic Coherence*

**Problem:** Generated itineraries were geographically incoherent and unrealistic

#### **Initial Issues:**

- LLM scheduled distant clusters on the same day (e.g., Mandai Zoo + Marina Bay + Sentosa)
- No respect for travel time between districts
- Activities ordered illogically (indoor morning, outdoor midday on hot days)
- Food recommendations disconnected from daily activity locations

#### **Debug Process:**

1. Reviewed user feedback showing itineraries requiring 4+ hours of daily transit
2. Analyzed LLM's raw outputs → No geographic constraints in prompts
3. Realized generic prompts lacked Singapore-specific planning heuristics

### **Solution:**

Created dedicated RAG document with planning rules

- Added `00\_planning\_rules\_and\_templates.md` to RAG corpus
- Defined explicit geographic clusters and travel time heuristics
- Specified weather-based activity placement rules

- Enforced "one district per day" constraint with optional nearby secondary district

### **Key Rules Enforced:**

#### *Geographic Clustering Rule (Most Important)*

- Each day **MUST** focus on **\*\*one primary district/cluster\*\***
- Maximum of **\*\*one nearby secondary district\*\*** may be added
- Do **NOT** schedule distant clusters on the same day

#### *Time Block Structure*

- Morning: outdoor or walking-heavy (if weather allows)
- Midday: indoor or shaded
- Afternoon: museums/cafes/shopping
- Evening: food + scenic views

#### *Travel Time Heuristic*

- Within same district: minimal transit
- Between districts: assume 30-45 minutes
- Avoid more than one cross-district move per day

### **Prompt Enhancement:**

*# In recommender.py system prompt:*

*"You **MUST** strictly follow the planning rules and heuristics provided in rag\_notes. These rules override general creativity. Use the geographic clusters defined in the rules to ensure feasible daily itineraries."*

**Result:** Itineraries became realistic, with activities naturally clustered by geography and time-of-day logic

**Lesson Learned:** Prompt engineering is highly iterative. Initial prompts were too vague, resulting in generic outputs. Adding domain-specific constraints via RAG documents dramatically improved output quality. RAG isn't just for Q&A—it's also a powerful mechanism for encoding business rules and heuristics.

## **6.3 Testing Tools & Debugging Approach**

### **Primary Debugging Tools:**

1. FastAPI Interactive Docs (`/docs`)
  - Directly tested API endpoints without frontend
  - Validated request/response schemas

- Rapid iteration on agent parameters
- 2. Streamlit Debug Expander
  - Real-time display of session state, tool outputs, and error traces
  - Collapsible section preserves UI cleanliness while providing deep inspection
  - Captured RAG notes, LLM prompts, and raw JSON responses
- 3. Structured Error Logging
  - Implemented `traceback.print_exc()` for all agent errors
  - Displayed full stack traces in UI during development
  - Helped identify exact failure points in multi-agent workflows
- 4. Incremental Testing Strategy
  - Started with single-agent testing (e.g., only RAG agent)
  - Progressively added agents to identify integration issues
  - Isolated LLM prompt changes to measure impact on output quality

## 6.4 Design Decisions & Trade-offs

Throughout development, several architectural and implementation choices required careful consideration of trade-offs:

### *Decision 1: Use Pre-loaded Documents instead of having user upload for RAG*

#### **Rationale:**

- Ensures quality and accuracy of recommendations
- Simplifies implementation (no file upload, parsing, validation)
- Appropriate for domain-specific application where expert curation adds value

**Trade-off:** Less flexibility for user-specific constraints or personalized content

**Reflection:** For a Singapore trip planner, curated documents provide higher quality than user-uploaded content. In other domains (e.g., corporate knowledge base), user uploads would be essential.

### *Decision 2: Use in-memory dictionary instead of database for session storage*

#### **Rationale:**

- Simpler implementation for prototype
- Fast access with zero latency
- Sufficient for single-server deployment during development

**Trade-off:**

- Sessions lost on server restart
- Not suitable for multi-server deployment (no shared state)
- No persistence for analytics or user history

### *Decision 3: Use GPT-4o-mini for all text generation tasks*

#### **Rationale:**

- Cost-effective (\$0.15 per 1M input tokens)
- Sufficient quality for itinerary generation
- Simplifies model management and prompt engineering

#### **Trade-off:**

- Could use larger model (GPT-4) for higher quality at 10x cost
- Could use different models for different tasks (e.g., smaller model for intake, larger for planning)

**Reflection:** GPT-4o-mini strikes an excellent balance of cost and quality for this use case. The quality difference from GPT-4 was minimal in testing.

### *Decision 4: Non-critical agents (Weather, RAG, SQL) fail gracefully without blocking itinerary generation*

#### **Rationale:**

- Weather/RAG/SQL enhance quality but aren't essential
- Better to provide degraded service than fail completely
- Improves system resilience

**Trade-off:** Increased complexity in error handling and result aggregation

**Reflection:** This decision significantly improved user experience. Early versions would crash entirely if any agent failed.

## 6.5 Future Improvements

1. User Authentication & Profiles
  - o Save trip history
  - o Store preferences for faster planning
  - o Share itineraries with friends
2. Export Functionality



- PDF export with itinerary + map
- Google Calendar integration
- Email delivery
- 3. Interactive Map Integration
  - Visual itinerary on Google Maps
  - See daily routes
  - Discover nearby attractions
- 4. Multi-Language Support
  - Translate interface and outputs
  - Support international travelers
- 5. Performance Optimization
  - Cache RAG results for common queries
  - Pre-compute embeddings for frequent searches
  - Implement request batching for parallel LLM calls